Original scientific paper

Informacije MIDEM
Journal of Microelectronics,
Electronic Components and Materials
Vol. 44, No. 4 (2014), 296 – 302

Implementation of Non-periodic Sampling True Random Number Generator on FPGA

Taner Tuncer[1], Erdinç Avaroğlu[2], Mustafa Türk[3], A.Bedri Ozer[1]

[1]Firat University, Department of Computer Engineering, Elaziğ, Turkey
[2]Inonu University, Department of Informatrics, Malatya, Turkey
[3]Firat University, Electrical Electronics Engineering Elaziğ, Turkey

Abstract: Random numbers are essentially required for various cryptographic applications. It is ideal to use nondeterministic random number generators in cryptography field since they are able to generate high-quality random numbers. In this paper, a Ring Oscillator (RO) based True Random Number Generator (TRNG) that can be used in cryptographic applications was developed. In this system, random numbers are generated by non-periodic sampling. Sinusoidal iterator with chaotic behavior was used for generation of non-periodic sampling signals. In TRNG system; three different scenarios, each of which contains three inverters, with 25, 10 and 5 RO circuits were implemented on FPGA environment. Randomness tests of numbers that are generated by TRNG with non-periodic sampling were carried on according to the NIST 800.22 test suit.The results have shown that the proposed system can be used in the cryptographic systems.

Keywords: Non-periodic Sampling, Ring Oscillator, Jitter, Statistical Test

Implementacija neperiodičnega vzorčnega generatorja naključnih števil na FPGA

Izvleček: Naključna števila so izrednega pomena pri številnih kriptografskih aplikacijah. Idealna je uporaba nedeterminističnih generatorjev naključnih števil, saj ti zagotavljajo visoko kvalitetna naključna števila. V članku je predstavljen generator naključnih števil (TRNG) na osnovi obročnega oscilatorja (RO). Naključna števila so v sistemu generirana z neperiodičnim vzorčenjem. Za generiranje neperiodičnih signalov vzorčenja je bil uporabljen sinusoidni iterator s kaotičnim obnašanjem. V TRNG sistemu so uporabljeni trije scenariji s po tremi razsmerniki s 25, 10 in 5 RO vezji v FPGA okolju. Test naključnosti števil je bil izveden na osnovi NIST 800.22 testnega procesa. Rezultati upravičujejo uporabnost predlaganega sistema v kriptografiji.

Ključne besede: neperiodično vzorčenje, obročni oscilator, statistični test

** Corresponding Author's e-mail: ttuncer,@firat.edu.tr*

1 Introduction

Random numbers are expected to have three essential features to be used in computer sciences. Firstly, generated numbers are to be unpredictable. Secondly, generated random numbers are required to have good statistical properties. Lastly, generated number streams should not repeat again. TRNG consists of three main parts as shown in Figure 1.1. These are Noise Source, Digitizer and Post-Processing. To generate random numbers, the first thing to do is to obtain a random signal from noise source. Thermal noise from a diode, jitter from a clock signal, metastability in a signal or mouse movements can be generally used as noise source. Discrete *s/i/* signal is obtained by sampling *n(t)* signal from noise source. When noise source is not ideal, *s/i/* signal does not have good statistical properties. Therefore, post-processing is preferred to improve the shortcoming of noise source or to obtain numbers with good statistical properties. After this stage, numbers are subject to statistical tests so as to determine whether they are random or not.

Generation of key in uncontrolled and unreliable environment negatively affects the system security of cryptographic applications. Thus, it is more appropriate to generate keys on hardware in terms of system security. It is getting more and more popular to generate keys on programmable hardware such as FPGA /1, 2, 3/. One

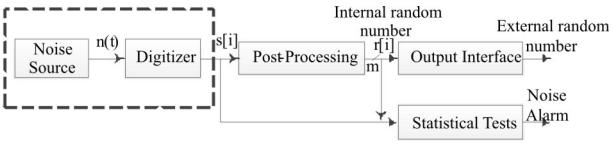

Figure 1.1: TRNG structure

of the most common methods to generate random numbers is to use Ring Oscillator structure achieved by inverters in FPGA /4/. There are 114 ROs and 13 inverters in each RO in TRNG system proposed by /4/. Post-processing is employed to improve statistical properties of generated random numbers. Random numbers were able to be generated at 2.5 Mbps data rate in this system. The drawback of this system is high power consumption. A new RO based structure was proposed by /1/ to overcome this drawback. In this structure, each RO has got a sampling unit. Number of ROs and number of inverters in each RO were decreased to 25 and 3 by using separate sampling units, respectively. This system manages to generate random number with good statistical properties at 100 Mbps since it does not include post-processing unit. It is suggested that post-processing unit is to be used in TRNG systems with cryptographic applications /5/. However, it is not ideal to use the system in /1/ for cryptographic applications since it does not have post-processing unit. In literature, basic processes such as XOR, Von Neumann corrector and LFSR (Linear Feedback Shift Register) Shuffler are used as post-processing /4, 6/. Although post-processing decreases data rate of TRNG, it is to be utilized in cryptographic applications /5/.

In addition to these approaches, random numbers can also be generated by Phase-Locked Loop hardware /6/. Usage of PLL (Phase-Locked Loop) in TRNG systems is rare due to the fact that PLL are limited in FPGA hardware. In another study, Open Loop Structure TRNG was proposed for usage of generated numbers in cryptographic applications /7/. The biggest advantage of this system is that it generates numbers at high data rates and does not require special components such as PLL. In /8/, random numbers could be generated at 20 Mbps by Open Loop Structure on FPGA. Besides, hybrid structures including usage of TRNG and PRNG (Pseudo Random Number Generator) were also proposed in literature /9, 10/. Structures such as Linear Feedback Shift Register (LFSR), chaotic map and cellular automata were used as PRNG /11, 12, 13, 14/. In /15/, the method and design of a pseudo random binary sequence generator operating at 10 Gb/s was implemented. In /16/, a PRNG was designed with usage of logistic map. The system has three logistic maps that are authenticated by different initialization vectors. The proposed system successfully passed statistical tests. Although PRNGs are not utilized in cryptographic applications, the system was analyzed against brute force and differential attacks. After all, the system was proved to be convenient for cryptographic applications. Sequential and parallel hybrid system was proposed by using LFSR structure in /17/. The system was developed on FPGA environment and it obtained successful results against cryptographic attacks.

In this paper, a RO based TRNG system was proposed for random number generation. The proposed system was implemented on FPGA for 3 different scenarios. Jitter, which is obtained from Ring Oscillators on FPGA, was used as noise source. Sampling of jitter was done by usage of non-periodic signals. Sinusoidal iterator, having chaotic behavior, was employed to get non-periodic signal. The system was implemented on FPGA hardware in real time, and statistical tests of generated numbers were conducted according to NIST 800.22 test suit.

Our contribution in this paper can be summarized as below.

It was shown that true random numbers generated by non-periodic sampling instead of periodic sampling can be used in cryptographic applications. A TRNG system that can operate in real time was developed. TRNG system could be turned into a more simple structure as statistical properties of numbers generated in three different scenarios were found to be good. Hence, the proposed TRNG structure is simpler than others in literature.

2 True random number generator

One of the most effective ways to generate random numbers is to use a noise source. Deviations of clock signals, which are generated in noise source, vary from their correct positions. This variability is called as jitter. Jitter, generated by clock signals, can be expressed by the Gaussian distribution as shown in Equation 2.1 /18/.

$$J_{RJ}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\left(\frac{x^2}{2\sigma^2}\right)} \quad (2.1)$$

Jitter is generally an undesired feature in a system. Due to being random, it is ideal to use jitter in systems such as TRNG. Figure 2.1 shows a jitter occurring in a clock signal.

TRNG systems need to contain two oscillators, one is fast and the other is low, as shown in Figure 2.2. While fast oscillator is obtained from a noise source, low oscillator can be a periodic or non-periodic signal. Clock signal from noise source is sampled in low oscillator by

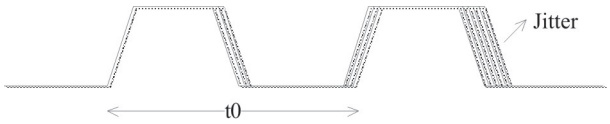

Figure 2.1: Jitter occurring in a clock signal

D flip flop. If the standard deviation in the period of the low oscillator is much greater than the fast oscillator, two consecutive samplings in the oscillator being sampled are regarded as uncorrelated /19/. In other words, numbers generated by sampling are going to be random.

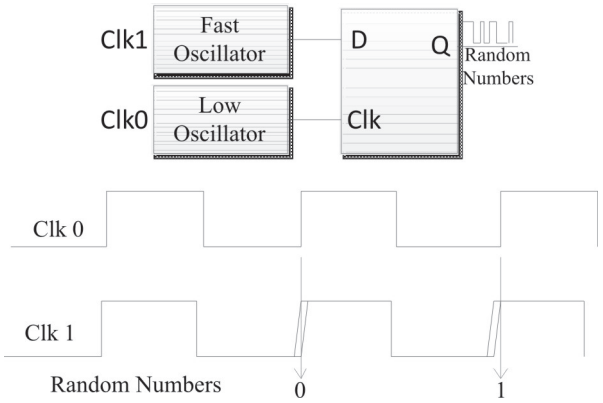

Figure 2.2: Random number generation

Usage of ROs is a classical method in TRNG. Quality of random numbers generated by a RO-based TRNG depends on three parameters below. These are:

- Number of oscillators in system
- Sampling frequency
- Number of inverters in oscillators

Randomness of numbers generated by TRNG is dependent on variability of jitter signal from each RO. Figure 2.3 shows structure of a simple RO.

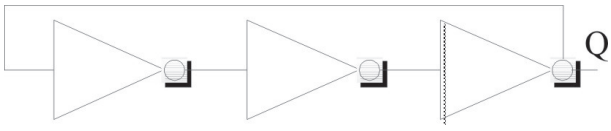

Figure 2.3: Structure of a simple RO

Frequency of signal from RO is as expressed in Equation 2.2. f_s is proportional to number of inverters (n) in system and delay time (t_{inv}) of inverter achieved by LE on FPGA.

$$f_s = \frac{1}{(2nt_{inv})} \tag{2.2}$$

Frequency of low oscillator is to be lower than f_s to be able to generate random numbers by RO-based TRNG.

3 Sinusoidal iterator

Functions with chaotic behaviors can be used as random number generators when given an initial value or seed value. Different number streams can be generated in a fast and easy way by changing initialization condition /20/. One of the functions is sinusoidal iterator given in Equation 3.1.

$$x_{n+1} = ax_n^2 \sin(\pi\, x_n) \tag{3.1}$$

The simplified form of Equation 3.1 for a= 2.3 and seed value x_0=0.7 is as shown in Equation 3.2. Figure 3.1 shows change of random number obtained after 1000 iterations for seed value x_0=0.7.

$$x_{n+1} = \sin(\pi\, x_n) \tag{3.2}$$

Figure 3.1: Change of random numbers obtained by sinusoidal iterator for seed value x_0=0.7

4 Non-periodic sampling TRNG

The TRNG system used in this paper is based on the system proposed by Knut Wold. Knut's system has 25 ring oscillators, each of which has 3 inverters. Instead of using post-processing, output of RO was sampled so that statistical deficiency caused by ROs could be eliminated. However, post-processing unit is advised to be included for cryptography applications. In this paper, non-periodic sampling TRNG system was experimented for three different scenarios. Firstly, non-periodic TRNG structure which includes 25 ROs with 3 inverters each was achieved. Statistical tests of random numbers generated by this structure were successful. In the following configurations, RO number was decreased to 10 and 5 ROs, respectively. Two non-periodic sampling TRNGs were developed on FPGA. Random numbers generated by systems in last two scenarios were also successful in NIST tests.

By this way, TRNG structure was simplified. Each scenario includes XOR post-processing unit. Sinusoidal iterator was used for non-periodic sampling in the system. The reason why Sinusoidal iterator is used instead of simpler structures such as Linear Congruential with chaotic behavior was that sinusoidal iterator has good randomness feature /21/. Figure 4.1 shows implementation of sinusoidal iterator on FPGA.

Random number generation was achieved by sinusoidal iterator, and thus non-periodic sampling signal was also obtained. Multiplication, sinus and comparison modules that can operate with floating point num-

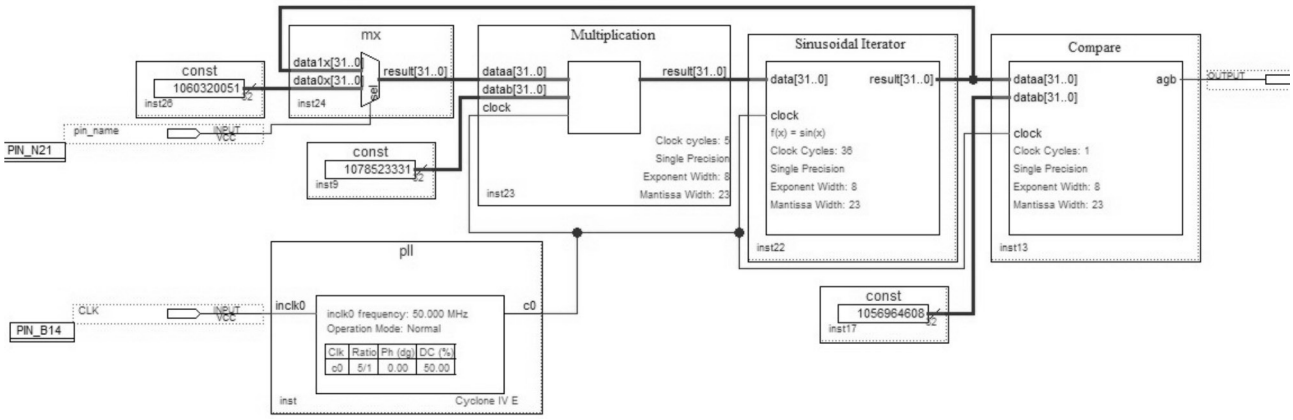

Figure 4.1: Implementation of sinusoidal iterator on FPGA

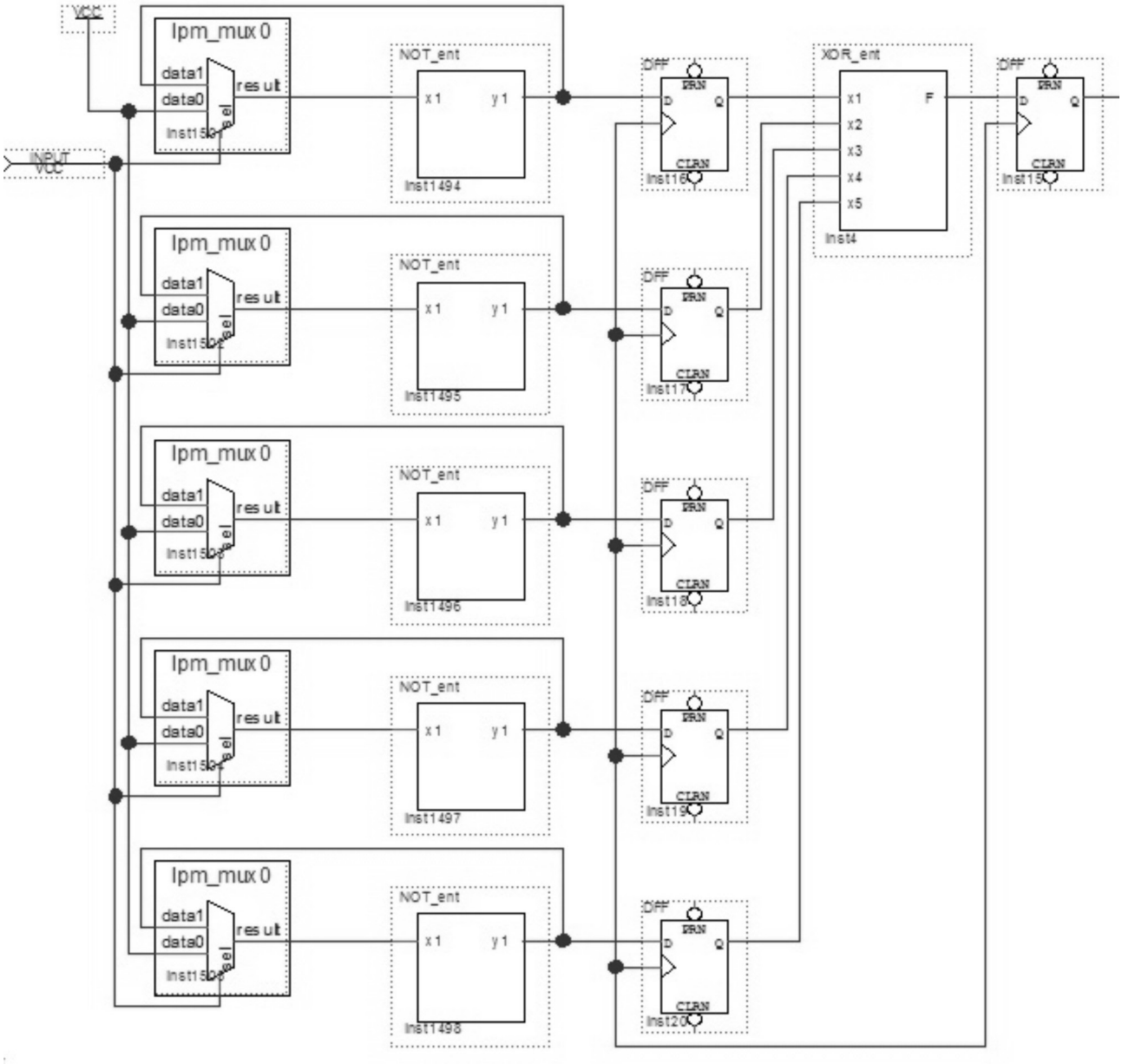

Figure 4.2: Implementation of TRNG with 5 ROs on FPGA

ber were used to achieve sinusoidal iterator. Besides, constant modules were also included so that mux and floating point number could be also used in the system. The value obtained as output, x_n, was provided as input value for the next iteration by mux so that the system could continuously generate random numbers. Initially, the system initialization value (x_0), the seed value in other words, was loaded. Const2 module was loaded by 0.7 or $(3f333333)_H$ in floating number expression. Const1 module was loaded by pi value or $(4048f5c3)_H$ in floating number expression. In order to represent floating point number generated by sinusoidal iterator as 0 and 1, floating number is compared with 0.5 or $(3f000000)_H$ in the output. The clock signal in the system is 200 MHz and the sampling signal obtained by sinusoidal iterator is generated once in 42 clocks. This duration is the sum of the executive times of multiplication (5 clocks), sinus (36 clocks) and comparator (1 clock) hardwares. The generated signal was fed to sampling system of TRNG as input. Figure 4.2 shows TRNG structure which consists of 5 ROs. A separate sampling unit was employed for each RO. Random numbers obtained as a result of sampling were once again sampled in XOR circuit and true random numbers were generated. Lastly, generated numbers were subject to post-processing to improve their statistical properties. In the end, numbers generated in real time were stored in a memory unit so that statistical tests could be applied on them. Figure 4.3 shows the memory unit, where numbers are stored to carry out statistical tests of numbers which are generated by non-periodic sampling TRNG.

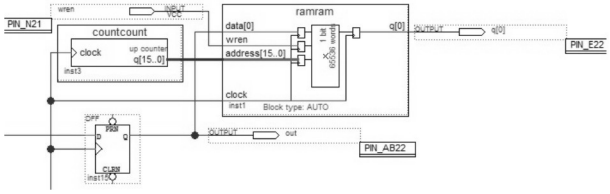

Figure 4.3: The memory unit

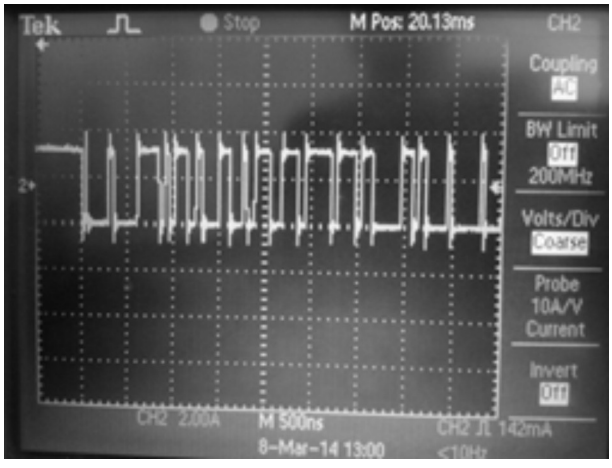

Figure 4.4: Random numbers generated in real time

The proposed structure was implemented in real time on Altera-based Cyclone IV FPGA. The change of random numbers generated by hybrid systemin real time was given in Figure 4.4.

5 Statistical tests and results

Statistical test suites play a very important role in assessing the randomness quality of numbers produced by random number generators. Several tests were developed to analyze randomness of numbers generated by generators. One of these tests is NIST test suit. There are totally 15 tests in NIST test suit, and parameters of each test are mentioned in /22/. The value α is one of the most important parameters used in tests and at the same time it is known as significance level. Determining significance level as 0.01 declares that randomness of numbers to be tested has 99% confidence value. Another parameter is P-value and known as measurement of randomness. If P-value equals to 1, numbers have perfectly random. If P-value equals to 0, then numbers are not random at all. Significance level, α, of numbers to be used in cryptographic applications must be chosen as an appropriate value. A test is assumed to be successful if P-value is equal to or greater than α value. Otherwise, test is considered to be unsuccessful and numbers are not random. Significance level is generally chosen between /0.001, 0.01/. The significance level in this paper was chosen as 0.01. The test was assumed to be successful in case the P-value obtained from each test is greater than 0.01. Results of P-value regarding to numbers generated by the system for 3 scenarios were given in Table 5.1. Numbers generated by the system successfully passed all of the tests as shown in the Table 5.1.

Table 5.1: NIST test results

Tests	P_{Value} (5 RO)	P_{Value} (10 RO)	P_{Value} 25RO
The Frequency (Monobit) Test	0,327	0,356	0,427
Frequency Test within a Block	0.446	0,412	0,598
The Runs Test	0.501	0,853	0,856
Tests for the Longest-Run-of-Ones in a Block,	0.751	0,652	0,840
The Binary Matrix Rank Test	0.398	0,742	0,873
The Discrete Fourier Transform (Spectral) Test	0.644	0,349	0,657
The Non-overlapping Template Matching Test	0.611	0,599	0,819
The Overlapping Template Matching Test	0.763	0,845	0,659

Maurer's "Universal Statistical" Test	0.730	0,945	0,783
The Linear Complexity Test	0.815	0,421	0,594
The Serial Test	0.494	0,584	0,688
The Approximate Entropy Test	0.437	0,746	0,841
The Cumulative Sums (Cusums) Test	0.506	0,355	0,322

Table 5.2 denotes number generation rates and the used ROs numbers of both proposed method and RO based TRNG known in the literature. The number generation rate of the proposed system is 4.77 Mbps and the number of ROs used is 5. According to Table 5.2, when compared our method with other TRNGs, it can be said that the proposed method is a more appropriate design. The number of ROs decreased from 25 to 5 since numbers generated by the system succeed in NIST tests. This is because the system generator numbers by non-periodic sampling and randomness is provided with respect to NIST tests. In spite of the above advantage, the system has a handicap; the power consuming of the system is high because of non-periodic sampling generator. As a remedy to this handicap, we propose the use of a LUT(Lookup Table) based hardware containing signals generated by sinusoidal iterator.

Table 5.2: Compare of both proposed method and RO based TRNG known in the literature.

TRNG	Rate (Mbps)	Number of ROs	Number of inverters in each RO
Multi oscillator rings (Sunar et al. /4/)	2.5	114	13
Multi oscillator rings (Wold et al. /1/)	100	25	3
Multi oscillator rings (Schellenkens at al./23/)	2.5	110	3
The proposed method	4,77	5	3

6 Conclusion

One of the most significant tasks in cryptographic systems is generation of keys in secured environments. Therefore, it is critical to generate keys by a random number generator on hardware such as FPGA. In this paper, non-periodic sampling TRNG system was implemented on FPGA in real time. NIST 800.22 test suit was applied on generated numbers to show that the developed system is suitable for cryptographic applications. Random numbers generated by 3 different scenarios were found out to attain good statistical properties. Another advantage of the proposed system is that the unpredictability of generated number streams was enhanced by non-periodic sampling unit. In other words, the system is more resistance against cryptographic attacks. The disadvantage of the system is that extra logic elements were required to implement sinusoidal iterator and the average data rate was 4.77 Mbps.

References

1. K. Wold, C.H. Tan, "Analysis and Enhancement of Random Number Generator in FPGA Based on Oscillator Ring", International Conference on Reconfigurable Computing and FPGAs, pp.385-390, 2008.
2. R. Santoro, O. Sentieys, S. Roy, "On-the Fly Evaluation of FPGA- Based True Random Number Generator", IEEE Computer Society Annual Symposium on VLSI, pp.55-60, 2009.
3. E. Erkek, T. Tuncer, "The Implementation of ASG and SG Random Number Generators", International Conference on System Science and Engineering(ICSSE), pp.363-367, 2013.
4. B. Sunar, W. J. Martin, D.R. Stison, "Aprovably Secure True Random Number Generator With Built-in Tolerance to Active Attacks", IEEE Transaction on Computers, Vol:56, No:1, pp.109-119, 2007.
5. O. Cret, T. Gyofi, A. Suciu, "Implementing True Random Number Generators Based on High Fanout Nets", Romanian Journal of Information Science and Technology Vol:15, No:3, p:277-298, 2012.
6. V. Fischer, M. Duratovsky, "True Random Number Generator Embedded in Reconfigurable Hardware", LNCS 2523, pp.415-430, 2003
7. J. Danger, S.Guilley, P. Hoogvorst, "High Speed True Random Number Generator Based on Open Loop Structures in FPGAs", Microelectronics Journal, Vol:4, p.1650-1656,2009
8. F. Lozac'h, M. Ben-Romdhane, T. Graba, J. Danger, "FPGA design of an Open-Loop True Random Number Generator", 16th Euromicro Conference on Digital System Design, 2013.
9. K.H. Tsoi, K.H. Leung, P.H.W. Leong, "Compact FPGA-based True and Pseudo Random Number Generators", Proceedings of the 11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, 2003.
10. N. M. Thamrin, G. Witjaksono, A. Nuruddin, M. S. Abdullah, "An Enhanced Hardware-based Hybrid Random Number Generator for Cryptosystem", International Conference on Information Management and Engineering, p:152-156, 2009.

11. T. Stojanovski, J. Pihl, L. Kocarev, "Chaos-Based Random Number Generators Part II: Practical Realization", IEEE Transaction on Circuits and System I: Fundemental Theory and Applications, Vol:48, No:3 pp.382-385, 2001.
12. M. Jonathan, C.J. Cerda, C.D. Martinez, H. David, K. Hoe, " Random Number Generators using Cellular Automata implemented on FPGAs", 44th IEEE Southeastern SymposiumSon system Theory, pp.67-72, 2012.
13. Sewak, K. Rajput, P. Panda, A.K., "FPGA implementation of 16 bit BBS and LFSR PN sequence generator: A comparative study," Electrical, Electronics and Computer Science (SCEECS), 2012 IEEE Students' Conference on , vol., no., pp.1,3, 1-2 March 2012.
14. I.Cicek, A.E. Pusane and G. Dundar, "A Novel Design Method for Discrete Time Chaos Based True Random Number Generators", Integration, The VLSI Journal, 2014.
15. L. Pavlovič, M. Vidmar and S. Tomažič, "10 Gb/s 215-1 pseudo-random binary sequence generator", Journal of Microelectronics, Electronic Components and Materials Vol. 42, No. 2 ,104–108, 2012.
16. M. Francois, T.Grosges, D. Barchiesi, R. Erra, "Pseudo Random Number Generator Based on Mixing of Three Chaotic Maps", Communications in Nonlinear Science and Numerical Simulation, Vol.19, No.4, p:887-895, 2014.
17. Thamrin, N.M.; Witjaksono, G.; Nuruddin, A.; Abdullah, M. S., "An Enhanced Hardware-based Hybrid Random Number Generator for Cryptosystem", International Conference on Information Management and Engineering, pp.152,156, 2009.
18. Petrie, C. S., Connelly, J. A., "A noise based IC random number generator for applications in cryptography", IEEE Trans.Circuit & Systems, Vol. 47, No. 5, p. 615-621, 2000.
19. B. Valtchanov, A. Aubert, F. Bernard, V. Fischer, Modeling and observing the jitter in ring oscillators implemented in FPGAs, in *11th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems,* p. 1–16, 2008.
20. S. Ergün, S. Özoğuz, "A chaos-modulated dual oscillator-based truly random number generator.", In Proceedings, International Symposium on Circuits and Systems, 2482–2485, 2007.
21. A. B. Ozer, "CIDE: Chaotically Initialized Differential Evolution", Expert Systems with Applications, Vol:37, Issue 6, p: 4632-4641, 2010.
22. NIST, "NIST Random Number Generation and Testing", 2006.
23. D. Schellekens, B. Preneel, I. Verbauwhede, "FPGA Vendor Agnostic True Random Number Generator", International Conference on Field Programmable Logic and Applications, p:1-6,2006

Arrived: 14. 05. 2014
Accepted: 14. 07. 2014